Internet Messaging

Wams, J.M.S.; van Steen, M.R.

published in
Practical Handbook of Internet Computing
2004

document version
Publisher's PDF, also known as Version of record

Contributors

0-8493-0052-5/00/$0.00+$.50

Contents

0-8493-0052-5/00/$0.00+$.50

1

Internet Messaging

Jan Mark Wams, Maarten van Steen
Vrije Universiteit Amsterdam

CONTENTS

1.1 Introduction

One can argue that messaging is the *raison d'être* of the Internet. For example, as of 2002, the number of active electronic mailboxes is estimated to be close to 1 billion and no less than 30 billion e-mail messages are sent *daily* with an estimated 60 billion by 2006. Independent of Internet messaging, one can also observe an explosion in the number of messages sent through the Short-Message Services (SMS): by the end of 2002, the number of these messages has been estimated to exceed 2 billion per day. Messaging has established itself as an important aspect of our daily lives, and one can expect that its role will only increase.

As messaging continues to grow, it becomes important to understand the underlying technology. Although e-mail is perhaps still the most widely applied instrument for messaging, other systems are rapidly gaining popularity, notably instant messaging. No doubt there will come a point at which users require that the various messaging systems are integrated, allowing communication to take place independent of specific protocols or devices. We can already observe such an integration of, for example, e-mail and short-messaging services through special gateways.

In this chapter, we describe how current Internet messaging systems work, but also pay attention to telephony-based messaging services (which we refer to as *telecom messaging*) as we expect these to be widely supported across the Internet in the near future. An important goal is to identify the key shortcomings of current systems and to outline potential improvements. To this end, we introduce a taxonomy by which we classify and compare current systems and from which we can derive the requirements for a unified messaging system.

1.2 Current Internet Solutions

We start by considering the most dominant messaging system in the Internet: electronic mail. We will discuss e-mail extensively and take it as a reference point for the other messaging systems.

0-8493-0052-5/00/$0.00+$.50

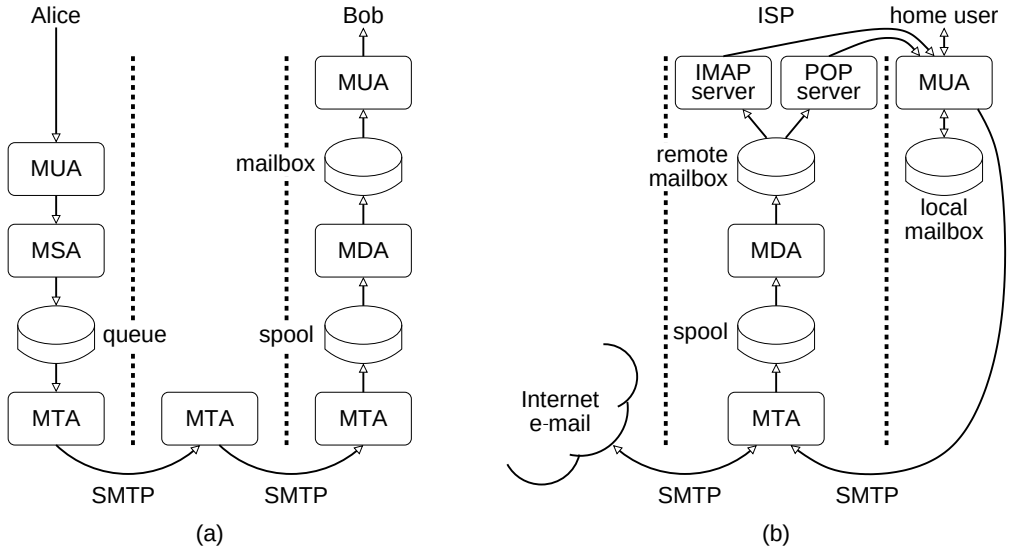

FIGURE 1.1
(a) The general organization of e-mail. (b) How e-mail is supported by an ISP.

These include network news (a bulletin-board service), and the increasingly popular instant messaging services. Our last example is Web logging, which, considering its functionality, can also be thought of as an Internet messaging service.

1.2.1 Electronic mail

Electronic mail (referred to as e-mail) is without doubt the most popular Internet messaging applications, although its popularity is rivaled by applications such as instant messaging and the telecom messaging systems that we discuss in Section 1.3. The basic model for e-mail is simple: a user sends an electronic message to one or more explicitly addressed recipients, where it is subsequently stored in the recipient's mailbox for further processing.* One of the main advantages of this model is that the asynchronous nature of communication: the recipient need not be online when a message is delivered to its mailbox, but instead, can read it at any convenient time.

Principal Operation

The basic organization of e-mail is shown in Figure 1.1(a) and consists of several components. From a user's perspective, a *mailbox* is conceptually the central component. A mailbox is simply a storage area that is used to hold messages that have been sent to a specific user. Each user generally has one or more mailboxes from which messages can be read and removed. The mailbox is accessed by means of a *mail user agent* (MUA), which is a management program that allows a user to, for example, edit, send, and receive messages.

Messages are composed by means of a user agent. To send a message, the user agent generally contacts a local *message submit agent* (MSA) that temporarily queues outgoing messages. A crucial component in e-mail systems is formed by *mail servers*, also referred to as *message transfer agents*

*It should be noted that many users actually have multiple mailboxes. For simplicity, we will often speak in terms of a single mailbox per recipient.

(MTAs). The MTA and the sender's site is responsible for removing messages that have been queued by the MSA, and transferring them to their destinations, possibly routing them across several other MTAs. At the receiving side, the MTA spools incoming messages, making them available for the *message delivery agent* (MDA). The latter is responsible for moving spooled messages into the proper mailboxes.

Assume that Alice at site *A* has sent a message *m* to Bob at site *B*. Initially, this message will be stored by the MSA at site *A*. When the message is eventually to be transferred, the MTA at site *A* will set up a connection to the MTA at site *B* and pass it message *m*. Upon its receipt, this MTA will store the message for the MDA at *B*, which, in turn, look up the mailbox for Bob to subsequently store *m*. In the Internet, mail servers are generally contacted by means of the *Simple Mail Transfer Protocol* (SMTP), which is specified in RFC 2821 [Klensin, 2001].

Note that this organization has a number of desirable properties. In the first place, if the mail server at the destination's site is currently unreachable, the MTA at the sender's site will simply keep the message queued as long as necessary. As a consequence, the actual burden of delivering a message in the presence of unreachable or unavailable mail servers is hidden from the e-mail users.

Another property is that a separate mail spooler allows for an easy *forwarding* of incoming messages. For example, several organizations provide a service that allows users to register a long-lived e-mail address. What is needed, however, is that a user also provides an actual e-mail address where incoming messages can be forwarded to. In the case of forwarding, a mail server simply passes an incoming message to the MSA, but now directed to the actual address.

Remote Access

The organization as sketched in Figure 1.1 assumes that the user agent has continuous (local) access to the mailbox. In many cases, this assumption does not hold. For example, many users have an e-mail account at an *Internet Service Provider* (ISP). In such cases, mail sent to a user is initially stored in the mailbox located at his ISP. To allow a user to access his mailbox, a special server is needed as shown in Figure 1.1(b).

The remote access server essentially operates as a proxy for the user agent. There are two models for its operation. In the first model, which has been adopted in the *Post Office Protocol* (POP3) described in RFC 2449 [Gellens et al., 1998], the remote access server transfers a newly arrived message to the user, who is then responsible for storing it locally. Although POP3 allows to keep a transferred message stored at the ISP, it is customary to configure user agents to instruct the server to delete any message that the agent had just fetched. This setup is often necessary due to the limited storage space that an ISP provides to each mailbox. However, even when storage space is not a problem, POP3 provides only minimal mailbox search facilities, making the model not very popular for managing messages.

As an alternative, there is also a model in which the access server does not normally delete messages after they have been transferred to the user. Instead, it is the ISP that takes responsibility for mailbox management. This model is supported by the *Internet Message Access Protocol* (IMAP), which is specified in RFC 2060 [Crispin, 1996]. In this case, the access server provides an interface that allows a user to browse, read, search, and maintain his mailbox. IMAP is particularly convenient for mobile users, and in principle can support even simple access devices such as wireless handhelds such as GSM cell phones (although special gateways are needed).

Naming

To enable the transfer of messages a scheme for addressing the source and destination is necessary. For Internet e-mail, an address consists of two parts: the name of the site to where a message needs to be sent, which, in turn, is prefixed by the name of the user for which it is intended. These two parts are separated by an at-sign ("@"). Given a name, the e-mail system should be able to set up a connection between the sending and receiving MTA to subsequently transfer a message, after which

```
; <<>> DiG 9.1.0 <<>> mx cs.vu.nl
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 43753
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 4, ADDITIONAL: 3

;; QUESTION SECTION:
;cs.vu.nl.                      IN      MX

;; ANSWER SECTION:
cs.vu.nl.               86069   IN      MX      1 tornado.cs.vu.nl.
cs.vu.nl.               86069   IN      MX      2 zephyr.cs.vu.nl.

;; ADDITIONAL SECTION:
tornado.cs.vu.nl.       86069   IN      A       192.31.231.152
zephyr.cs.vu.nl.        86069   IN      A       192.31.231.66
```

FIGURE 1.2
Response to a DNS query using the dig tool (edited).

it can be stored in the addressed user's mailbox. In other words, what is required is that an e-mail name can be *resolved* to the network address of the destination mail server.

Resolving an e-mail name requires support from the Internet *Domain Name System* (DNS) [Mockapetris, 1987; Albitz and Liu, 1998]. Consider sending an e-mail to an address johndoe@cs.vu.nl. In this example, johndoe identifies the user at site cs.vu.nl. To send a message, it is necessary to identify a mail server that can handle incoming e-mail traffic. For Internet e-mail, DNS allows to store such information in what are known as *mail exchange* records, or simply MX records. For example, using a program called *domain information groper* (dig), a DNS query requesting an MX record for cs.vu.nl returns the answer shown in Figure 1.2.

The most important part of the response is the answer section (shown in boldface in Figure 1.2), which states that there are two mail servers for cs.vu.nl. The preferred mail server is named tornado.cs.vu.nl, while a secondary server named zephyr.cs.vu.nl is also available. To initiate an SMTP session, the sender's MTA usually sets up a TCP connection to the preferred MTA at the destination site, for which it needs the server's IP address. In our example, this would require resolving the name tornado.cs.vu.nl, which is, in principle, done by means of another DNS query. DNS anticipates such additional queries when asked for an MX record and includes an *additional section* containing the IP addresses of the returned mail servers thus avoiding another query.

Once the message has been transferred to the destination MTA, it is the task of the latter to resolve the user name that is part of the e-mail address to the appropriate mailbox. How this user-name resolution is done is not prescribed by SMTP.

Message Formats: MIME

An important issue in any messaging system is that sender and receiver agree on the format of the message content. Such an agreement is possible by including the description of that format as part of the message header. This is the principle underlying *Multipurpose Internet Mail Extensions* (MIME), which we briefly discuss next.

An e-mail message is a string of values that is mapped into readable text by a character set. The best known character set is the North American ASCII set, which has 96 characters, but lacks European characters such "ß," "å," "ç," and "ñ." Moreover in Asian, Russian, Arabic, and other languages, totally different word and character sets are in use. MIME is a standard defined to accommodate these different sets, as well as graphics, sound, and encodings like HTML (see, e.g., RFC 2231 [Freed and Moore, 1997]). The MIME standard is not exclusively used for e-mail, other Internet messaging systems, like netnews, use it too.

```
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 8bit
```

FIGURE 1.3
MIME header example: plain text message.

```
MIME-Version: 1.0
Content-Type: text/html; charset=big5
Content-Transfer-Encoding: base64
```

FIGURE 1.4
MIME header example: A HTML/Chinese (Big5) message.

An e-mail message is formed after a memo: it has a *body* with the actual message and a *header* containing information about the author, date of creation, subject, and so on. MIME specifies a number of header fields, which are column-separated keyword-value pairs, to define the structure of the message body. For example, the header fields in Figure 1.3 describe that the message body is composed—and should be displayed—using ASCII-text mapping. Figure 1.4 indicates that the body is to be interpreted as a Chinese HTML message that is (base64) encoded for transfer.

Another useful MIME content type is *multipart* to indicate that a message that multiple body parts that are separated by a unique string.

There are four multipart subtypes. The most common subtype is *mixed* to indicate a series of generic body parts that carry their own header-fields. The are other subtypes like *alternative* for representing the same data in different formats like plain text and HTML, and *parallel* for parts that need to be simultaneously processed like sound and graphics. Body parts are bracketed by lines with only the unique string and usually have their own MIME header fields, as shown in Figure 1.5.

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=unique-135711

--unique-135711
Content-type: text/plain; charset=US-ASCII

Look at this picture.
--unique-135711
Content-Type: image/jpeg; name=picture.jpg
Content-Transfer-Encoding: base64
Content-Disposition: inline

P9jJ4AAQSk ... DBkSEw8UHRof
      ⋮
UKJgkjhUUOTaOGZs5wP712Q==
--unique-135711
Content-type: text/enriched

<bold><italic>BYE!</italic></bold>
--unique-135711--
```

FIGURE 1.5
Multipart message example.

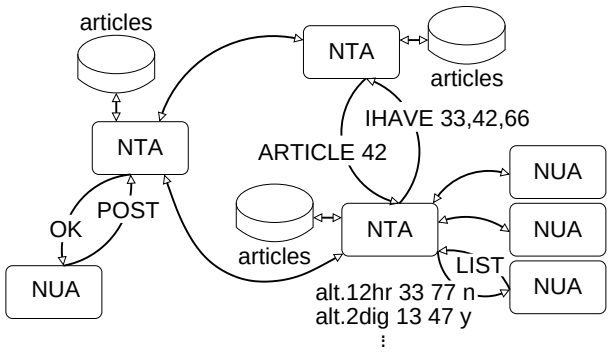

FIGURE 1.6
The general organization of network news.

1.2.2 Network News

Network news, also abbreviated to *netnews*, gained its popularity as part of USENET, a logical network mainly consisting of many computers that used simple dialup phone lines for message transfer. The netnews model is that of an electronic bulletin board: messages are put up on the board to be read and reacted to by others. In netnews, messages are referred to as *articles* that are *posted* in a specific *newsgroup*. A newsgroup is thus a collection of logically related articles and forms the electronic representation of a bulletin board. A non-technical overview of network news is given by Comer [2000].

A user provides the netnews system with the name of a newsgroup in order to read articles. The *header* of any new article that has not yet been read by the user is then transferred to the user. If the user wants to read the entire article, he will request for the transfer of the article's *body*. After reading an article, a user can respond by posting a reaction in that same newsgroup (which again appears as just another article). Cross postings by which an article refers to an article in a different newsgroup is also possible.

Note that users do not actively delete articles. However, to prevent that postings consume storage indefinitely, system administrators generally remove articles after some time. Unlike e-mail where messages are permanently stored until explicitly deleted by a recipient, this policy makes news articles impermanent unless special measures are taken to store them permanently.

Principal Operation

The core of the network news system is formed by a huge collection of news servers that are spread across thousands of different sites. A news server, also referred to a *News Transfer Agent* (NTA), is capable of receiving, sending, and storing articles. The basic organization is shown in Figure 1.6. A client, called a *News User Agent* (NUA), connects to a news server to read and post articles for one or several newsgroups. Likewise, servers connect to each other to exchange articles as we discuss in more detail below. Although the figure suggests a principal difference between clients and servers, no such difference actually exists. In fact, the protocol that is used between a client and server and the one used between two servers is the same. All information exchange follows the *Network News Transfer Protocol* (NNTP), specified in RFC 977 [Kantor and Lapsley, 1986].

A news server should connect to one or more existing *news feeds*, which are just other news servers that are willing to exchange articles. In many cases, a news feed is operated by a separate organization such as an ISP. When a news server contacts a news feed, it requests the transfer of new articles. There are several operations available to establish such a transfer of which some important ones are listed in Table 1.1.

Operation	Description
LIST	Returns a list of newsgroups available at the callee with each entry identifying the first and last article in that group.
GROUP	Makes a specified group "current," and returns an estimate of the number of articles at the callee in that group.
ARTICLE	Transfers (to the caller) a specified article in the current group.
POST	Tells the callee that an article has been posted at the caller.
IHAVE	Tells the callee that a specific article is available to be sent.
NEWNEWS	Returns a list of articles that have been posted at the callee in specific news groups.
NEWGROUPS	Returns a list of newsgroups that have been created at the callee.

Commonly used operations to establish the transfer of articles between two news programs.

The transfer protocol is relatively simple and has not been changed since its specification in 1986. However, practice has shown that extensions and deviations from the original specification were needed. In particular, the communication between servers, and that between a client and a server are different enough to warrant further refinements, effectively leading to two very similar, yet different protocols. These refinements are described in RFC 2980 [Barber, 2000].

Naming

An important difference between netnews and e-mail is that there is no need to explicitly name and lookup news servers. Instead, the address of a news feed is assumed to be known at the time a news client or server is configured so that its address can be readily used to setup an NNTP session. Jointly, these sessions ensure that articles are *flooded* through the network consisting of netnews servers. In contrast, for e-mail it is necessary to devise a naming scheme by which users and mail servers can be looked up at runtime. This naming scheme is needed to support the *point-to-point* communication in e-mail systems.

Naming in news therefore restricts itself to newsgroups, and implicitly also articles. In particular, it is important to have a suitable naming scheme for the tens of thousands of newsgroups that currently exist. To this end, a hierarchical naming scheme has been devised that is simple, yet flexible enough to support a large number of newsgroups. A newsgroup name is a series of strings separated by a dot, such as comp.os.research. In this example, comp identifies the broad category of newsgroups related to computer science, which is further divided into newsgroups dealing with operating systems (os) and, in particular, the one containing articles on research in this area (research).

Each article has a unique identifier consisting of two parts separated separated by the at-sign. An example of such an identifier is 3e1ed38c$1@news.cs.vu.nl (see also RFC 1036 [Horton and Adams, 1987]). The second part identifies the host where the article was first entered into the news system, in this example news.cs.vu.nl. The first part is a unique identifier normally generated by the host named in the second part (and hidden for the user). In principle, an article's identifier is globally unique and is never reused: it is a so-called true identifier [Wieringa and de Jonge, 1995].

1.2.3 Instant Messaging

One of the upcoming means of user-centric communication across the Internet is instant messaging. The model underlying instant messaging is that of *synchronous communication*: a message can

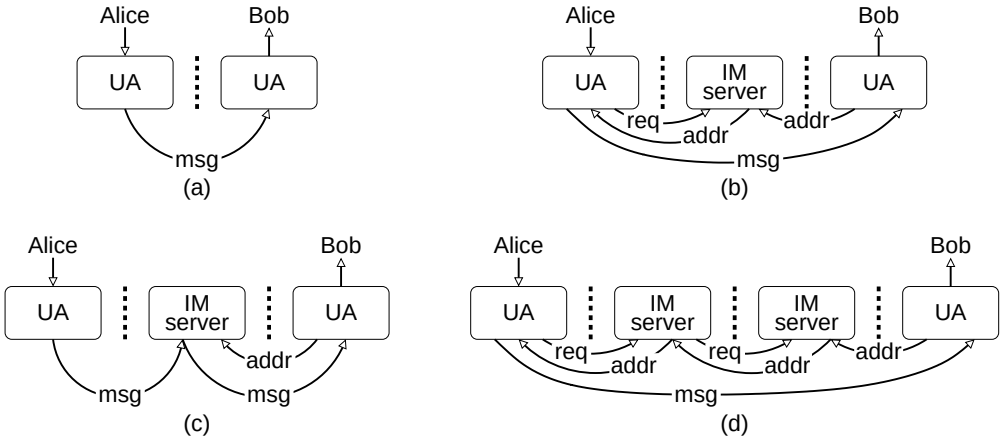

FIGURE 1.7
Setting up an instant-messaging connection: (a) directly, (b) through a central server, (c) centralized, including messaging, and (d) through different servers.

be successfully transferred only if the destination is willing to receive it at the time it is sent. In many other respects, instant messaging strongly resembles e-mail and the two forms are sometimes integrated into a single system. One of the first one-on-one instant messaging system was called "term-talk" an ran on the Plato system as early as 1973. One of the earliest full blown instant messaging systems appeared in M.I.T.'s Athena system [Belville, 1990]. Instant messaging on the Internet originated as *Internet Relay Chat* (IRC, described in RFC 1459 [Oikarinen and Reed, 1993] and updated in RFCs 2810-2813 [Kalt, 2000]), but became really popular with the introduction of ICQ (pronounced as "I seek you"). Currently, there are many instant messaging clients, whereas instant messaging services are provided by large organizations such as AOL and Microsoft.

An instant messaging system generally has a separate component, called a *presence information service*, that is used to inform users of each other's presence (see also RFC 2778 [Day et al., 2000b]). Such a service allows a user to see whether it is possible to send a message to someone else. When a user logs in, his presence is published and forwarded to subscribers of that information. Likewise, a user can indicate that he is temporarily not reachable, or has logged out. Managing presence information is increasingly becoming an important issue as it strongly affects the privacy of publishers. We return to presence information in more detail below.

Principal Operation

The principal operation of an instant messaging service is quite simple. In all cases, we need to first set up a *channel* between the communicating parties. Let us consider the case that Alice wants to communicate with Bob. If Alice has Bob's address then, in principle, she can setup a channel directly to Bob's user agent (UA) as shown in Figure 1.7(a).

The main drawback of this approach is that Bob's contact address must be fixed (i.e., the address where Alice can reach Bob), but also that Alice can setup unsolicited channels to Bob. To alleviate these problems, many instant messaging services adopt the scheme shown in Figure 1.7(b). In this case, a central server keeps track of online clients (whose contact address may be different each time they come online). Alice sends a setup request to the server who subsequently returns Bob's address, possibly after checking whether Alice is authorized to setup a channel to Bob. Note that the central server may also be used as an intermediary for *all* communication between Alice and Bob, that is, including the instant messages sent between them, as shown in Figure 1.7(c).

Status	Alert	Description
OFFLINE	No	No instant-messaging client is currently running at the recipient.
ONLINE	Yes	The recipient's instant-messaging client is currently running.
AWAY	Yes	The recipient's client is running, but invitations cannot be accepted.
BUSY	No	The recipient's client is running, but invitations are not notified.

Examples of different states maintained by a presences information service. The column *alert* indicates whether the recipient is notified when a setup request arrives.

The obvious drawback of the central server is that it forms a potential bottleneck. This centralized approach, even when multiple servers are used, is recognized as one of the main scalability problems in IRC. To circumvent these scalability problems, several instant messaging servers can be used, as shown in Figure 1.7(d). In this solution, Alice contacts a local instant-messaging server and requests a communication channel to Bob. Her local server then contacts a server that controls connections to Bob, and requests a communication endpoint to Bob's client. After the proper security checks have been made and Alice is indeed found to be authorized to contact Bob, Bob's address is returned to allow the setup of a connection.

This use of a distributed instant messaging service scales well, as only the servers that are local to Alice and Bob need to assist in setting up a connection. However, it also introduces a lookup problem, because the server local to Alice needs to locate Bob's local instant messaging server. A simple solution, but one that has not been widely deployed yet, is to follow the same approach as in e-mail. In principle, every site makes use of a single instant messaging server and users simply identify themselves by their e-mail address. Assume Bob's e-mail address is bob@cs.vu.nl. When Alice wants to contact Bob, her (well known) local instant messaging server queries the Domain Name System (DNS) for the instant messaging server at cs.vu.nl, analogous to asking DNS for the name of the mail server at that site. Once its name is returned, its IP address can then be looked up using DNS again.

So far, we have assumed that instant messaging takes place only between pairs of individuals. In general, this need not be the case. Two different forms of multi-party instant messaging exist. First, setting up a connection between two parties can easily be extended by inviting another party, leading to an *ad hoc group*, or *chat session*. In this case, each message is sent to all members participating in the session. An invited party can *join* the session, and any joined party can later *leave* again. A session dissolves when the last member leaves.

The second type of multi-party instant messaging is through so-called *chat rooms*, which are effectively permanent sessions. To enter a chat room, a user needs to setup a connection to a well-known server that handles all communication for that chat room, effectively leading to the communication scheme shown in Figure 1.7(c). Each message sent to the server is multicast to every other client that has entered the chat room. Unlike ad hoc groups, chat rooms continue to exist even after the last member has left. By their nature, a chat room is useful for online discussions on a very specific subject, and this is indeed the way that they are generally organized.

Presence Information Service

As we mentioned, an important component of an instant messaging service is a service that provides presence information. In a minimalist approach, such a service merely reports whether a user is online or offline, allowing an initiator to see whether it makes sense to even try to setup an instant messaging connection. However, presence information can, and often is, extended with other possible states, as shown in Table 1.2.

Many variations on these states exist. For example, some presence services automatically switch a recipient from *online* to *away* when there has been no interaction with the instant-messaging client for some while. Likewise, when an invitation is sent out to a recipient who is currently *busy*, the inviting client may receive a message telling that the other other party does not want to be disturbed.

It is not difficult to see that where instant messaging by itself is relatively simple, a presence information service can easily grow into a sophisticated and complex part of an instant messaging service. Following the general architecture as described in RFC 2778 [Day et al., 2000b], a presence information service may also provide the means to send *notifications* when a client's status changes. Such a notification may be useful, for example, when Alice wants to contact Bob as soon as he can accept invitations again.

Despite its attractiveness, the real problem with this functionality starts when thinking about security. In effect, Alice *subscribes* to notifications concerning state changes of Bob. Although it may seem obvious that Bob should be in full control of permissible subscriptions, practice shows that this is not always the case. However, as also laid down in RFC 2779 [Day et al., 2000a], a client should always be in full control concerning who is allowed to send instant messages, and who is allowed to subscribe to presence-state changes. This model has been adopted by the IETF working group on *Extensible Messaging and Presence Protocol* (XMPP).

In essence, before Alice can subscribe to presence information concerning Bob, XMPP requires that she sends Bob a request for subscription. If this request is granted, Bob can pass her the appropriate credentials by which she can obtain a subscription at the presence information service. Bob, in turn, can always request the presence service to unsubscribe Alice.

How simple this model may seem, it has severe implications for the design and implementation of a presence information service. The simplest situation is when the presence service is implemented as a single (trusted) centralized server. In that case, managing subscriptions boils down to checking lists of subscribers and sending notifications as needed. However, when dealing with a distributed presence information service, we are essentially dealing with the same problems that general publish/subscribe systems have. Having to manage many users who may be geographically widely dispersed, scalability problems suddenly become paramount and obvious solutions do not exist (see, e.g., [Carzaniga et al., 2001]). Relatively little is known on developing large-scale wide-area notification systems, let alone when having to face various security attacks.

Naming

Naming is generally straightforward in instant-messaging systems and mainly concerns identifying users. For this reason, systems are gradually adopting the e-mail naming scheme. In the case of chat rooms, instant-messaging service providers generally offer a list of topics for which a chat room is hosted. By selecting a topic, a user is then allowed to join a chat session. Naming in such cases is therefore implicit and of of less importance than with core instant messaging.

However, naming in many popular instant-messaging systems is still much of nuisance. In particular, several systems such as ICQ simply provide a unique (long) number that is to be used as ID. The drawback of using these numbers is similar to using network addresses instead of host names: they are difficult to remember by humans. To circumvent problems, users simply build local lists of aliases for those people who are regularly contacted.

1.2.4 Web Logging

As a last example of Internet messaging, we briefly consider an increasingly popular form known as Web logging, or simply "blogging" [Blood, 2002]. A Web log (or "blog") can be viewed as a unidirectional form of messaging: a user simply maintains a log of messages that others can generally only read. However, the tools that make it possible for readers to react to messages is growing rapidly. Web logging can be considered analogous to columns and commentaries in news

papers. As the popularity of this form of messaging continues to increase, communities of similar logs are starting to grow in which loggers (or "bloggers" as they are normally called) are referring and reacting to each other's work. In a sense, we are seeing a phenomenon that resembles the use of network news.

There is a lot of Web logging going on, but because most Web logs are self-published, precise numbers are difficult to obtain. Experts, however, agree that Web logging is growing ever more popular, probably due to the existence of tools and sites that allow easy entry. Many companies have an in-house Web log, millions of private Web logs exist on the WWW, and there are some high-volume Web logs that serve a huge number of readers.

One of the bigger sites that allows everybody with a Web browser to easily maintain a Web log, is Blogger (blogger.com). According to Blogger, over a million people have used their service to start a Web log, and subscriptions show an exponential growth. There are a few big and influential web-logs. Over the course of 2002, the Web log Drudge (drudgereport.com) served around a billion pages. The Web log Slashdot (slashdot.org), for example, also serves millions of pages per day with "news for nerds." Needless to say that Slashdot has multiple editors, and that sophisticated distributed moderating takes place to keep this huge volume usable.

Principal Operation

The principal operation of Web logging is extremely simple: a user simply publishes material on a single site that can be read by anyone accessing that site. Many tools are available that ease the process of updating and managing published material, effectively hiding the technical intricacies related to Web servers.

An important difference with all messaging systems discussed so far, is that there are, in principle, no recipients. All material related to Web logging is conceptually published at a single site that needs to be polled regularly if a reader wants to keep track of changes. Alternatively, some systems already offer subscription facilities by which readers are automatically notified when updates occur. In practice, update notifications are simply sent by means of e-mail.

1.3 Telecom Messaging

Internet messaging is rivaled in popularity by telecom messaging. The two are becoming increasingly intertwined and there is no doubt that they will be fully integrated at some point in the future. Traditionally, telecom messaging was implemented *in-band*, that is, making direct use of voice channels. Examples of in-band telecom messaging are the fax system, bulletin board systems, and the French Minitel. A notable exception to in-band telecom messaging is paging, which usually has a dedicated radio frequency channel. The introduction of the global standard for telecommunications, namely *Signaling System number Seven* (SS7), allowed for out-of-band handling of data.

Principal Operation

Both the in-band and out-of-band telecom messaging rely on the SS7 protocol stack. ITU defined SS7 as a packet-switching four-layer stack resembling the seven layer ISO/OSI network stack with the top four layers integrated into one (for an overview of SS7, see Dreher and Harte [2002]). Data packets are sent over the SS7 network to setup (and tear down) voice connections. Routing information is stored in databases called *Home Location Registers* (HLRs).

From the perspective of messaging, it is interesting to note that also (limited sized) data packets can be sent over the SS7 network, just like IP-packets can be sent over the Internet. This allows

for the implementation of efficient messaging schemes like the Short Message Service (SMS), and which we briefly discuss below.

Naming

For telecom messaging, names take the form of telephone numbers. Traditionally, these numbers have been directly used for routing. Dialing 1234 would get a phone connected to exit four of exit three of exit two of exit one of the exchange office that the telephone was connected to. Nowadays, most telephone numbers have three basic parts: a *country code*, an *area code*, and a *subscriber number*. For local calls, the first two parts need not be explicitly provided.

Partly due to cellular phones, routing schemes needed to be adjusted and affected the way naming was deployed. To facilitate a fixed number for these roaming devices, the area code in a telephone number was used to also designate a particular cell-phone operator, effectively diverging from the geographical interpretation initially tied to area codes. This approach has nowadays been taken another step further, as the original area code is now also used to designate different types of services, such as toll-free calls, premium-rate calls, normal cell phones, pagers, and so on.

A simple aliasing scheme is used to avoid that users need to remember numbers, by associating several letters to a single digit. This may lead to telephone numbers such as 555-shoe-shine which actually stands for the number 555-7463-74463.

1.3.1 Short Message Service

One particular popular telecom messaging system in Europe and Asia is the Short Message Service (SMS). As we mentioned, already by the end of 2002 more than 2 billion SMS messages were sent daily, and this number is still rapidly increasing. As an aside, in Japan the more advanced i-mode messaging, is similarly popular.

Besides SMS there is the *Enhanced Messaging Service* (EMS), which combines multiple SMSes into one EMS, and the (technically unrelated) *Multimedia Messaging Service* (MMS), which can handle much larger messages. Both EMS and MMS can currently (early 2003) not rival SMS in volume. There is one important drawback to SMS that is related to its use of SS7: it can carry only very short messages—around 160 characters long, in Europe. There are more characters in an SMS message, but some are used to store the callers telephone number and other data. On the other hand, SMS messages can be sent independent of voice traffic. So even if the network is loaded up to the point where it becomes impossible to make voice calls, SMS messages can still get through. The SMS system is a store-and-forward system, allowing SMS messages to be delayed if necessary. This makes SMS a very robust service.

As an interesting side note, SMS messages originally were never intended for subscriber to subscriber usage. They were designed for voice-mail notification.

Principal Operation

SMS can be seen as a service that allows the transfer of a set of characters implemented in the emphMobile Application Part layer (MAP) on top of SS7. There are two similar standards, the American IS-41 (or ANSI-41) and the international GSM-MAP.

SMS messages are routed by what are known as SMS centers (SMSC). An SMS center sends an SMS request to the addressed HLR to find the–usually roaming–recipient. The HLR has two possible responses. In case of an *inactive* response, the recipient is currently off-line. The HLR will send an *active* response when the subscriber becomes available.

The SMS center tries to forward the SMS message to active, that is, online subscribers and receives a reply message stating whether or not the SMS message was successfully delivered. An SMS center keeps trying to send an SMS messages for a limited time. In the end it always sends back a report to the original sender of the SMS message, stating success or failure.

Dimension	Values
Time	immediate, impermanent, permanent
Direction	simplex, duplex
Audience	group, world
Address	single, list, all

The four dimensions of a taxonomy for comparing messaging systems.

Nowadays, SMS centers are connected to many systems and networks like fax, e-mail, voice-mail, WWW, IP networks, and so on. It is feasible to send SMSes from–and often to–a wired phone, web-page, PDAs, satellite phone, and so on.

Naming

Telecom messaging, by definition, uses telephone numbers to name the recipient and SMS is no exception. However, since the SMS centers are interconnected to many non-telecom messaging systems, several extensions have been proposed to allow cross-system sending.

1.3.2 SMS Cross Messaging

Two ways to address a non-telecom messaging system are currently popular, *address prefixing* and *keywording*. With address prefixing, the first part of the SMS message is reserved for the address of the recipient's naming scheme and is sent to a gateway that has a telephone number. For example, as SMS message such as "johndoe@cs.vu.nl Dinner at 8?" may be addressed to telephone recipient 8008, which acts as an e-mail gateway, delivering the message "Dinner at 8?" to the e-mail box of johndoe@cs.vu.nl.

With keywording, the telephone number of a *service* is used in combination with a keyword-like message. For example, the message "weather adam" would signal to a service provider that it should send an SMS message with the weather forecast for Amsterdam back to the sender.

Often a special short (four digit) telephone number is used in conjunction with address prefixing or keywording. Given the shortage of four digit telephone numbers, it is common for gateways and service providers to share a single short telephone number at which they can offer several services. It could be argued that prefixes and keywords actually become part of the naming and addressing scheme used in SMS cross messaging.

1.4 A Comparison

To compare the various messaging systems we have developed a simple taxonomy. This taxonomy is organized along the four most important aspects from the perspective of a user, as opposed to a technical or design perspective. With this taxonomy, any messaging system can be scaled with respect to four independent dimensions, which are shown in Table 1.3.

A messaging system can have one of three values in the time dimension: (1) *immediate*, meaning that all messages are short lived or available only once during a short period, (2) *impermanent* meaning that all messages are available pending their expiration or revocation by some set of rules, and (3) *permanent* meaning that all messages are available indefinitely unless a message is explicitly

Dimension	E-mail	News	Web log	IM/1-1	IM/group	SMS
Time	permanent	impermanent	permanent	immediate	immediate	permanent
Direction	simplex	duplex	duplex	simplex	duplex	simplex
Audience	world	group	world	group	group	world
Address	list	all	all	list	all	single

Classification of current messaging systems.

revoked by an authorized user.

With respect to direction, we distinguish two values. The value *simplex* means that a *write only* storage or channel is used for sending a message. The recipient cannot use the same storage or channel to reply. A reply has to be directed towards another storage or channel. The value *duplex* means that one store or channel is used for both reading and writing.

The audience describes the set of *potential* recipients of messages. We distinguish the following two values: (1) *world*, which stands for every user that has the hardware, software, and connectivity to use the system, and (2) *group*, which stands for a true subset of all potential recipients. In a grouped messaging system, users cannot send messages to a user outside their audience, even though this outsider is ready for any message and uses the same system. Restriction of audience (grouping) can be the result of restrictions related to the infrastructure or implementation. The system can also limit the audience as a service, security measure, or due to politics.

In the address dimension a messaging system can have three values: (1) *single*, if the system allows only one recipient per message, (2) *list* if the system allows for addressing more than one explicitly addressed recipient (i.e., a list of singles), and (3) *all* if the system allows for some form of broadcasting. Note that when all members of an audience are addressed, this does not mean that every addressed recipient will necessarily receive each message. As we already saw for Web logging, a system may require an addressee to explicitly *fetch* a message from storage or a channel.

The four dimensions are truly independent, although not all of the 36 combinations are equally useful. Note that it is easy to confuse audience and address: both are subsets of recipients. The audience comes with the system, and users have no direct influence on it. The address, on the other hand, is something the user determines and the system cannot influence. The intersection of audience and address is the set of recipients that will receive the message.

Using this taxonomy, we can easily compare the messaging systems discussed so far. In Table 1.4 we show how e-mail, network news, instant messaging, web logging, and short messaging can be classified to this taxonomy.

The easiest classification is that for e-mail: messages are kept in the system until explicitly destroyed. Also, it should be clear that e-mail employs unidirectional communication, while, in principle, there are no limitations concerning to whom a message can be sent. E-mail supports both single as well as multiple-addressed messages.

In the network news systems, articles are normally removed after some time. Special archives are used to permanently store messages, but these do not form an intrinsic part of news. Communication can be considered bidirectional as messages can be written to and read from the same channel. The targeted audience is always a group, namely the subscribers to a specific newsgroup. Posting an article is always to all subscribers, which effectively implies broadcasting.

Web logging messages generally have an impermanent status as they are regularly updated. However, many sites keep old messages available, giving them a permanent status. Initially, Web logging made use of unidirectional storage. However, modern systems provide an interface to allow readers

Medium	1996	1997	1998	1999	2000	2001
Diskette	74%	88%	67%	39%	7%	1%
Internet e-mail	9%	26%	32%	56%	87%	83%
Other Internet	12%	24%	14%	16%	2%	20%
Other	15%	7%	5%	9%	2%	1%
A virus can have more than one medium, so totals can exceed 100%.						

Changes in virus distribution media over time.

to post comments at the logger's site. This facility essentially turns Web logging into a duplex system. Clearly, there are no restrictions on who can read the logs, meaning that the targeted audience is the entire world. That the system also employs broadcasting is because everyone from the targeted audience can actually access the logs. Note that broadcasting is *implemented* through polling. As a result, if a reader does not access the logger's site, the addressed recipient will not be able to read the message.

We need to divide instant messaging into two groups. When considering the one-to-one way of messaging, it is clear that messages have an immediate character: they are never stored. Likewise, communication is unidirectional, with a targeted group that is, or rather should be, restricted by the service provider. As in e-mail, the addressed recipients are always a single user. When dealing with chat sessions, there are two major differences. First, communication is now essentially duplex: the same channel is used to send and receive messages. Also, messages are always addressed to the entire group of members that participate in the chat session.

Finally, the short messaging systems as supported by telephony-based infrastructures provide permanent messages (that can have a expire date). Messaging is, as in e-mail, unidirectional, just as the targeted audience can be anyone. Finally, messages in these systems are addressed to a single recipient.

1.5 A note on unsolicited messaging

Control concerning the receipt of messages has been briefly touched upon in the discussion so far. Although it is not our goal to go into these matters in great detail, there is at least one issue that needs to be addressed in this chapter: unsolicited messaging. Notably the e-mail system is known for its great "potential" to send unsolicited messages to its users. In the following, let us take a closer look at two forms of such messaging: the spreading of viruses and spamming.

1.5.1 Spreading viruses

The e-mail system has become the primary habitat of viruses, as concluded by a survey conducted in 2001 by ICSA Laboratories [Bridwell and Tippett, 2001] (see Table 1.5). This stands to reason since e-mail offers the largest homogeneous audience, but there is more that makes e-mail the number one target.

The primary success factor of most viruses is *impersonation*. With impersonation, a virus uses

the victim's identity and messaging address list to spread itself to the next collection of gullible users taken in by the apparent trustworthy origin of the message. This works especially well with e-mail, where users generally trust the apparent origin.

Furthermore, there are two strategies that work particularly well in combination with e-mail. In the case of *self-replication* a program is sent as part of the message and which handles its own replication after being delivered to the attacked messaging client. The other strategy is *coercion*, by which a message (often called a *hoax*) is voluntarily forwarded by the attacked user, simply because it solicits proliferation

The success of the self-replicating e-mail virus is largely due to the predominance of Microsoft's e-mail client "Outlook." The mere popularity of this product makes it attractive to exploit its security vulnerabilities so that by targeting Outlook, a virus maximizes the number of potential victims. There have been two successful variants of self-replicating viruses. *Worms* exploit bugs in the attacked messaging client to replicate as soon as the victim reads the message. *Trojans* form malicious software (often called *malware*) that poses as a picture or other harmless data that starts replicating when the victim opens the attachment.

E-mail facilitates multi-recipient messages, making it easy, cheap and effortless to send a single message to many users. Therefore e-mail also offers an attractive attack medium for hoaxes. Two main indicators that an e-mail is a hoax are the forwarding request and the lack of a date. Basically, a hoax is an e-mail version of the traditional chain letter, rumor, pyramid game or Ponzi scheme. A hoax traditionally travels without a malware payload and is usually platform independent. A hoax can be tenacious because well-intentioned users keep refurbishing it and the most successful variants thrive on.

1.5.2 Spam

E-mail seems very prone to spam, also called junk-mail or unsolicited commercial e-mail (UCE). Other messaging systems as the ones discussed can expect a similar fate. This situation will not change any time soon, because most spam is actually very effective. There is a simple reason for this effectiveness. Even though the vast majority of spam receivers simply discard the incoming spam messages, a small fraction actually does react, either willingly or accidently. Considering the size of the targeted group of recipients, and the small amount of money involved in reaching a group through e-mail, any small response is already enough to warrant success for the sender.

Another related cause for the effectiveness of spam, is that its senders actually hope for complaints by recipients. Each time a complaint comes in that identifies the recipient, the spammer will be left with a messaging address that is known to be read (a so-called active account). This information can be sold for a much higher price that addresses of unverified accounts.

At the moment of writing, spam is largely distributed through e-mail, but spam has since long been a problem for the USENET News system. It is now also becoming an issue for telecom and instant messaging services.

1.5.3 Protection mechanisms

In several countries, anti-spam laws are active or under consideration. However, it is doubtful whether any legislation will help. First of all, it will not help against *acquaintance spam*, that is, spam following a solicited message. Second, many messaging systems can be fooled easily, making it hard to track down the perpetrator.

Getting rid of unwanted messages like viruses and spam is generally difficult. In principle, unwanted messages can be filtered out at any hop a message makes. For e-mail there are three logical moments to delete unwanted messages: when it is sent, during message transfer, and when it is to be delivered. Removing (or marking) of unwanted messages is referred to as *filtering*.

In principle, filtering is simple. A message is checked against a list of *signatures*, which are known (nearly unique) combinations of bytes, and subsequently deleted, or marked, if the signature is found in the message. For example, if a message contains a word referring to a specific commercial product, chances are it is spam so that it should be deleted.

The problem with filters, however, is that there is always a fraction of false positives, that is, messages that are deleted as spam while, in fact, they are not. Likewise, filters will also lead to a fraction of false negatives, being spam messages that are not recognized as such. Practice shows that keeping both fractions small is difficult. Effective filtering will delete most spam messages (and likewise almost all viruses), while occasionally mistaking a message for spam, but almost never wrongly identify a virus. Note that filters need to be updated regularly for recognizing new viruses.

Filtering on signatures requires careful construction of the signatures to minimize false positives. Most messaging systems allow filtering rules to be hand crafted, but this approach is often not very effective. As an alternative, signature-delete-rules can also be constructed by specialists and subsequently downloaded (often by means of a paid subscription). This approach has traditionally been used for virus filtering mainly because it is the only way a virus' signature can be spread quicker than the virus itself. More sophisticated rules would still be needed to catch what is known as a *polymorphic virus*, that is, a virus that changes itself to avoid detection. The *blacklist filter*, which is also usually downloaded, is a signature-delete rule that filters on the senders address. This type of rule has been successfully used against spam in the past. However, spam has evolved to evade this type filtering rendering it increasingly less useful.

A more sophisticated approach is to generate rules based on statistical comparison of the content of previous unwanted messages. Generated rules, because of their adaptive property, have been proven to be very successful against various new types of spam that could not be detected using blacklists or other static signatures.

1.6 Toward unified messaging

Given that so many different messaging systems exist, it is not surprising that several attempts are being made toward their integration into a unifying system. Unification is generally interpreted as integration of the existing systems such that users can send and retrieve any type of message using only a single interface. Proposals range from relatively simple integration, such as described by Yeo et al. [2000], to advanced architectures that actually integrate many Internet and telecom services [Wang et al., 2000].

What many unified messaging systems do is actually concentrate on the integration of *technology* rather than the integration of messaging *models*. The result is often that a single messaging technology is used as the nexus for all other systems. E-mail often plays such a role. However, rather than placing technology as the key integrator, it can be argued that integration of messaging models is the key issue. This approach has essentially been adopted in, for example, the Mobile People Architecture [Maniatis et al., 1999].

Continuing to follow the user's perspective, unified messaging is more about models than technology. A relatively simple model that can cover all four dimensions from our taxonomy is the following. In this model, which we refer to as the *Unified Messaging* (UM) model, each message is said to be *targeted* to a specific user or a group of users. For simplicity, we assume that each message is *immutable* (i.e., it cannot be changed after being sent), and that it is usual short. These properties lead us to use the term *Targeted Immutable Short Message* or TISM for short. We use the name *target* to denote the destination of a TISM.

Any UM model should address the following requirements:

1. *Large-scale messaging*. Any model should be able to deal with handling hundreds of billions of messages per day between billions of users.

2. *Independence of trusted sites*: A UM model can be implemented using a client-server system with a trusted server, but also a peer-to-peer communication model [Milojicic et al., 2002], or even combination of both.

3. *Orthogonality of the four dimensions*. Every UM model should allow any combination of time, direction, audience, and address.

4. *Prevention of spam*. Each model must offer maximum control to prevent unsolicited messages, without restricting the freedom of speech.

Especially the prevention of unsolicited messages should be adequately dealt with. The following UM model, described by Wams and Van Steen [2003], does exactly that. A target protects each TISM with public key encryption and a *Message Authentication Code* (MAC) [Schneier, 1996]. In this model each target is associated with a unique *post-key*/*read-key* pair. To post a TISM, the proper post-key is needed. Likewise, to read a TISM, the proper read-key is needed. Without a read-key, it is sufficiently hard to reconstruct a TISM, even if a post-key and a copy of the encrypted TISM are available. Without a post-key, it is very hard to spoof a TISM even if the read-key is available. The unified messaging system (UMS) implementing this UM model will generate a post-key/read-key pair for every new target.

A target is identified by a systemwide unique binary string. We define a *target-ID* as this unique binary string. A (post-key, read-key, target-ID) tuple is denoted as *post/read-tuple*. Likewise we use *read-tuple* and *post-tuple*. When the UMS creates a target for a user, the user is returned a post/read-tuple, from which a separate post-tuple and read-tuple can be created. Typically, a user might create a target and distribute its read-tuple to others, enabling them to get the encoded TISMs from the target (using the target-ID), and to decode those TISMs (with the read-key). This is similar to a Web-log messaging system. Had the user distributed the post-tuple, an e-mail like system would have resulted.

The UMS user has a number of ways to distribute (key, target-ID) tuples. A user could pass on a tuple wrapped in a TISM. Alternatively, she could distribute a tuple through the World Wide Web or some other generic distribution system, or could store a tuple in a (local) name space system specially designed for the UMS. Other lookup models are also feasible.

To utilize the fine-grained control the UMS offers, the user needs a separate target for each different communication partner or group. This may sound complex, especially to users that manage all their Internet e-mail from one mailbox. However, most e-mail users already have many submailboxes. Likewise, most instant-messaging systems allow users to create any channel/room they want to. As another example, every netnews user can create a new `alt.*` group at will (like the actually existing `alt.swedish.chef.bork.bork.bork`). Creating a new box or channel, in one of these legacy messaging systems is limited by the ability to create a new entry in the accompanying name space. For example, finding a meaningful name for a new `alt.*` news group that does not already exist, is hard, as is the case for instant-messaging channels/rooms. Moreover, the e-mail submailboxes are usually not publicly addressable.

1.7 Outlook

Considering the popularity of e-mail and other messaging systems, and the convergence of Internet and telecommunications, it is beyond doubt that we can expect to see an exponential growth in network-based messaging in the near future. Unified messaging, in any form, will play a crucial role. Users will demand a simple messaging model that is independent from the devices they use for composing, sending, and receiving messages.

Another important observation is that messaging will further integrate with the many Web-based information systems. Message-based ordering is already common practice, but seamless integration of messaging, electronic commerce and information services can be expected. Again, unified messaging will show to be crucial.

An interesting development in this context is moving user agents toward Web servers, effectively offering end users no more than just a simple interface to messaging operations that are carried out at a remote server. This approach is already being taken in Web-based e-mail, which allows users to access their mailbox from any place provided they have access to the Web. Using Web clients as the universal means to access messaging systems, implies that unification and integration must take place at the server side. It is yet unclear whether this approach will succeed if we simply integrate technologies instead of unifying messaging models.

References

P. Albitz and C. Liu. *DNS and BIND*. O'Reilly & Associates, Sebastopol, CA, 3rd edition, 1998.

S. Barber. Common NNTP Extensions. RFC 2980, October 2000.

S. Belville. *Zephyr on Athena*. MIT, Cambridge, MA, February 1990.

R. Blood. *The Weblog Handbook*. Perseus Publishing, Cambridge, MA, 2002.

L. Bridwell and P. Tippett. ICSA Labs 7th Annual Computer Virus Prevalence Survey, 2001.

A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and Evaluation of a Wide-Area Event Notification Service. *ACM Trans. Comp. Syst.*, 19(3):332–383, August 2001.

D. Comer. *The Internet Book*. Prentice Hall, Upper Saddle River, NJ, 3rd edition, 2000.

M. Crispin. Internet Message Access Protocol - Version 4rev1. RFC 2060, December 1996.

M. Day, S. Aggarwal, G. Mohr, and J. Vincent. Instant Messaging / Presence Protocol Requirements. RFC 2779, February 2000a.

M. Day, J. Rosenberg, and H. Sugano. A Model for Presence and Instant Messaging. RFC 2778, February 2000b.

R. Dreher and L. Harte. *Signaling System 7 Basics*. APDG Publishing, Fuquay-Varina, NC, 2nd edition, 2002.

N. Freed and K. Moore. MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations. RFC 2231, November 1997.

R. Gellens, C. Newman, and L. Lundblade. POP3 Extension Mechanism. RFC 2449, November 1998.

M. Horton and R. Adams. Standard for Interchange of USENET Messages. RFC 1036, December 1987.

C. Kalt. Internet Relay Chat. RFCs 2810–2813, April 2000.

B. Kantor and P. Lapsley. Network News Transfer Protocol: A Proposed Standard for the Stream-Based Transmission of News. RFC 977, February 1986.

J. Klensin. Simple Mail Transfer Protocol. RFC 2821, April 2001.

P. Maniatis, M. Roussopoulos, E. Swierk, K. Lai, G. Appenzeller, X. Zhao, and M. Baker. The Mobile People Architecture. *ACM Mobile Comput. Commun. Rev.*, 3(3):36–42, July 1999.

D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to-Peer Computing. Technical Report HPL-2002-57, Hewlett Packard Laboratories, Palo Alto, CA, March 2002.

P. Mockapetris. Domain Names - Concepts and Facilities. RFC 1034, November 1987.

J. Oikarinen and D. Reed. Internet Relay Chat Protocol. RFC 1459, May 1993.

0-8493-0052-5/00/$0.00+$.50

B. Schneier. *Applied Cryptography*. John Wiley, New York, 2nd edition, 1996.

J.M.S. Wams and M. van Steen. Pervasive Messaging. In *First Int'l Conf. Pervasive Computing and Communications (PerCom)*, Los Alamitos, CA, March 2003. IEEE, IEEE Computer Society Press.

H. J. Wang, B. Raman, C. Chuah, R. Biswas, R. Gummadi, B. Hohlt, X. Hong, E. Kiciman, Z. Mao, J. S. Shih, L. Subramanian, B. Y. Zhao, A. D. Joseph, , and R. H. Katz. ICEBERG: An Internet-core Network Architecture for Integrated Communications. *IEEE Pers. Commun.*, 7(4):10–19, August 2000.

R. Wieringa and W. de Jonge. Object Identifiers, Keys, and Surrogates–Object Identifiers Revisited. *Theory and Practice of Object Systems*, 1(2):101–114, 1995.

C. K. Yeo, S. C. Hui, I. Y. Soon, and G. Manik. Unified Messaging : A System for the Internet. *Int'l J. Comp., Internet, and Mgmt.*, 10(3), September 2000.

Index